A. Appendix

- **SR-1**: [PReserve] **Must Not be Provided If** [No Conflicting Paths Res Others = No]
- **SR-2**: [PCancel] **Must Not be Provided If** [Path State = Occ]
- **SR-3**: [PRelease] **Must Not be Provided If** [Path State = Occ]
- **SR-4**: [BReserve] **Must Not be Provided If** [Track Block Mode = Res]
- **SR-5**: [BCancel] **Must Not be Provided If** [Track Block Mode = Res; Track Block State = Occ]
- **SR-6**: [BRelease] **Must Not be Provided If**[Track Block Mode = Res; Track Block State = Occ]
- **SR-7**: [BRelease] **Must Not be Provided If** [Track Block Mode = Res; Track Block State = Free]
- **SR-8**: [PReserve] **Must Not be Provided If** [No Blocks Are Reserved = No]
- **SR-9**: [PReserve] **Must Not be Provided If** [All Blocks Are Reserved = No]
- **SR-10**: [PCancel] **Must Not be Provided If** [No Blocks Are Occupied = No]
- **SR-11**: [PRelease] **Must Not be Provided If** [No Blocks Are Occupied = No]
- **SR-12**: [Switch] Must Not be Provided **If** [Track Block Mode = Res; Track Block State = Free; Point Position = Ali]
- **SR-13**: [Switch] Must be Provided **If** [Track Block Mode = Res; Track Block State = Free; Point Position = Una]
- **SR-14**: [Switch] Must be Provided **If** [[Track Block Mode = Res; Track Block State = Occ]
- **SR-15**: [Lock] Must be Provided **If** [Point Position = Ali; Point Mode = Unl]
- **SR-16**: [Lock] Must be Provided **If** [Point Position = Una; Point Mode = Unl]
- **SR-17**: [Unlock] Must Not be Provided **If** [Track Block Mode = Res; Track Block State = Free; Point Mode = Lock]
- **SR-18**: [Unlock] Must Not be Provided **If** [Track Block Mode = Res; Track Block State = Occ; Point Position = Ali; Point Mode = Lock]
- **SR-19**: [PReserve] Must Not be Provided **If** [All Points Are Aligned Correctly = No]
- **SR-20**: [PReserve] Must Not be Provided **If** [All Points Are Locked = No]
- **SR-21**: [Green] **Must Not be Provided If** [No Blocks Are Reserved = No]
- **SR-22**: [Green] **Must Not be Provided If** [No Blocks Are Res Others = No]
- **SR-23**: [Green] **Must Not be Provided If** [All Blocks Are Res Oneself = No]
- **SR-24**: [Green] **Must Not be Provided If** [All Points Are Aligned Correctly = No]
- **SR-25**: [Green] **Must Not be Provided If** [All Points Are Locked = No]
- **SR-26**: [Red] **Must Not be Provided** [too late] **If** [No Blocks Are Occupied = No; Signal State = Green]
- **SR-27**: [Red] **Must be Provided If** [No Blocks Are Occupied = No; Signal State = Green]
- **SR-28**: [Red] **Must Not be Provided** [too late] **If** [Route Mode = Can; Signal State = Green]
- **SR-29**: [Red] **Must be Provided If** [Route Mode = Can; Signal State = Green]
- **SR-30**: [Unlock] **Must Not be Provided If** [Signal State = Green]
- **SR-31**: [Switch] **Must Not be Provided If** [Signal State = Green]
- **SR-32**: [BCancel] **Must Not be Provided If** [Signal State = Green]
- **SR-33**: [BRelease] **Must Not be Provided If** [Signal State = Green]

B. Appendix

B.1. Initial Model

B.1.1. Context

```
CONTEXT SR0_Path
SETS
 PATH
 ROUTE
CONSTANTS
 PathConflict
 Route2InitPath
 PathSub
 NullPath
AXIOMS
 axm1: PathConflict ∈ PATH ↔ PATH
 axm2: ∀p·p∈PATH ⇒ (PathConflict ∩ id = ∅)
 axm3: ∀p·p∈PATH ⇒ (PathConflict = PathConflict∼)
 axm4: ∀p·p∈PATH ⇒ (finite(PATH))
 axm5: Route2InitPath ∈ ROUTE → PATH
 axm6: PathSub ∈ PATH ↔ PATH
 axm7: ∀p·p∈PATH ⇒ (∀p1,p2·p1 ∈ PathSub[{p2}] ∧ p ∉
     PathConflict[{p2}] ⇒ p∉PathConflict[{p1}])
 axm8: ∀p·p∈PATH ⇒NullPath∈ PathSub[{p}]
 axm9: NullPath ∈ PATH
END
```

B.1.2. Machine

```
MACHINE SR_M0
SEES SR0_Path
VARIABLES
 Route_Req
 Route_Cel
 Route2Path
 Route_Occ
INVARIANTS
 inv1: Route_Req ⊆ ROUTE
 inv2: Route_Cel ⊆ ROUTE
 inv3: Route2Path ∈ ROUTE ⇸ PATH
 inv4: Route_Occ ⊆ ROUTE
 inv5: ∀r1,r2·(r1 ≠ r2 ∧ r1 ∈ dom(Route2Path) ∧ r2 ∈ dom(
     Route2Path)) ⇒ (PathConflict~[Route2Path[{r1}]] ∩
     Route2Path[{r2}] = ∅)
 inv6: ∀r·r∈Route_Occ ⇒ (Route2Path[{r}] ≠ ∅)
 INITIALISATION:
 THEN
  act1: Route_Req := ∅
  act2: Route_Cel := ∅
  act3: Route_Occ := ∅
  act4: Route2Path := ∅
 END

 ATS_Request:
 ANY r
 WHERE
  grd1: r ∉ Route_Req
 THEN
  act1: Route_Req := Route_Req ∪ {r}
 END

 ATS_Cancel:
 ANY r
 WHERE
  grd1: r ∈ Route_Req
 THEN
  act1: Route_Cel := Route_Cel ∪ {r}
 END

 Route_Reserve:
 ANY r
 WHERE
  grd1: r ∈ Route_Req
  grd2: r ∉ Route_Cel
  grd3: r ∉ dom(Route2Path)
 THEN
  act1: Route2Path := Route2Path ∪ {r↦Route2InitPath(r)}
 END

 Route_Cancel:
 ANY r
 WHERE
  grd1: r ∈ Route_Req
  grd2: r ∈ Route_Cel
  grd3: r ∈ dom(Route2Path)
 THEN
  act1: Route_Req := Route_Req \ {r}
  act2: Route_Cel := Route_Cel \ {r}
  act3: Route2Path := {r} ⩤ Route2Path
 END

 Route_Release:
 ANY r
 WHERE
  grd1: r ∈ Route_Req
  grd2: r ∈ dom(Route2Path)
  grd3: Route2Path(r)=NullPath
 THEN
  act1: Route_Req := Route_Req \ {r}
  act2: Route2Path := {r} ⩤ Route2Path
 END

 Train_Enter:
 ANY r
 WHERE
  grd1: r ∈ dom(Route2Path)
  grd2: r ∉ Route_Occ
 THEN
  act1: Route_Occ := Route_Occ ∪ {r}
 END

 Train_move:
 ANY r cp sp
 WHERE
  grd1: r ∈ Route_Occ
  grd2: r ∈ dom(Route2Path)
  grd3: cp ∈ Route2Path(r)
  grd4: cp ≠ NullPath
  grd5: sp ∈ PathSub[{cp}]
 THEN
  act1: Route2Path(r) := sp
 END

 Train_Leave:
 ANY r
 WHERE
  grd1: r ∈ Route_Occ
  grd2: r ∈ dom(Route2Path)
  grd2: Route2Path(r) = NullPath
 THEN
  act1: Route_Occ := Route_Occ \ {r}
 END
```

B.2. First Iteration

B.2.1. Context

```
CONTEXT SR1_Block
EXTENDS SR0_Path
SETS
 BLOCK
CONSTANTS
 Path2Block
 PathReduce
 PathIncrease
AXIOMS
 axm1: Path2Block ∈ PATH ↔ BLOCK
 axm2: ∀p·p ∈ PATH⇒(∀q·q ∉ PathConflict[{p}]⇔(Path2Block
     [{p}]∩Path2Block[{q}]=∅))
 axm3: PathReduce ∈ (PATH\{NullPath})→(BLOCK→PATH)
 axm4: ∀p·p ∈ PATH\{NullPath}⇒(∃b·b∈BLOCK⇒
     PathReduce(p)(b)∈ PathSub[{p}])
 axm5: ∀p·p ∈ PATH\{NullPath}⇒(∃b·b∈BLOCK⇒Path2Block
     [{PathReduce(p)(b)}]=Path2Block[{p}]\{b})
 axm6: PPath2Block[{NullPath}]=∅
 axm7: PathIncrease ∈ PATH → (BLOCK → PATH)
 axm8: ∀p·p ∈ PATH ⇒(∃b·b∈BLOCK ⇒ p ∈ PathSub[{
     PathIncrease(p)(b)}])
 axm9: ∀p·p ∈ PATH ⇒(∃b·b∈BLOCK ⇒ Path2Block[{p}] ∪ {b}
      = Path2Block[{PathIncrease(p)(b)}])
END
```

B.2.2. Machine

```
MACHINE SR_M1
REFINES SR_M0
SEES SR1_Block
VARIABLES
```

```
...
Block2Route
Route2OccPath
INVARIANTS
 inv1: Block2Route ∈ BLOCK ⇸ ROUTE
 inv2: Route2OccPath ∈ ROUTE ⇸ PATH
 inv3: ∀p·p ∈ PATH ⇒ (∀q·q ∉ PathConflict[{p}] ⇔ (
     Path2Block[{p}] ∩ Path2Block[{q}] = ∅))
 INITIALISATION:
 THEN
  ...
  act5: Block2Route := ∅
  act6: Route2OccPath := ∅
 END

 Block_Reserve:
 ANY r p
 WHERE
  grd1: r ∈ Route_Req
  grd2: r ∉ Route_Cel
  grd3: r ∉ dom(Route2Path)
  grd4: p = Route2InitPath(r)
 THEN
  act1: Block2Route := Block2Route ∪ {Path2Block(p) ↦ r}
 END

 Block_Cancel:
 ANY r
 WHERE
  grd1: r ∈ Route_Cel
  grd2: r ∈ ran(Block2Route)
 THEN
  act1: Block2Route := Block2Route ⩥ {r}
 END

 Block_Release:
 ANY r b
 WHERE
  grd1:r ∈ Route_Occ
  grd2: b ∈ Block2Route∼[{r}]
  THEN
  act1:Block2Route := {b} ⩤ Block2Route
 END

 Train_Enter:
 REFINES Train_Enter
 ANY r
 WHERE
   ...
   grd3:r ∉ dom(Route2OccPath)
 THEN
   ...
  act2: Route2OccPath := Route2OccPath ∪ {r ↦ NullPath}
 END

 Train_Head_Move:
 ANY r op b
 WHERE
   grd1:r ∈ Route_Occ
   grd2:r ∈ dom(Route2Path)
   grd3:b ∈ Path2Block[{Route2Path(r)}]
   grd4:r ∈ dom(Route2OccPath)
   grd5: op = Route2OccPath(r)
   grd6: b ∉ Path2Block[{op}]
 THEN
  act1: Route2OccPath(r) := PathIncrease(op)(b)
 END

 Train_Rear_Move:
 REFINES Train_Move
 ANY r p sp op b
 WHERE
   ...
   grd6:r ∈ dom(Route2OccPath)
   grd7:op = Route2OccPath(r)
   grd8:b ∈ Path2Block[{op}] ∩ Path2Block[{p}]
   grd9:op ≠ NullPath
   grd10:sp = PathReduce(p)(b)
   grd11:PathReduce(op)(b) = NullPath ⇒ card(Path2Block[{p
       }])≠1
 THEN
  ...
  act2:Route2OccPath(r) := PathReduce(op)(b)
 END

 Train_Leave:
 REFINES Train_Leave
 ANY r
  ...
  grd4:Route2OccPath(r)==NullPath
 THEN
  ...
  act2:Route2OccPath := {r} ⩤ Route2OccPath
 END
```

B.3. Second Iteration

B.3.1. Context

```
CONTEXT SR2_Point
EXTENDS SR1_Block
SETS
 POS
 MODE
CONSTANTS
 POINT
 Point2InitPos
 Route2Point2Pos
 Lock
 Unlock
 Plus
 Minus
AXIOMS
 axm1: POINT ⊆ BLOCK
 axm2: Route2Point2Pos ∈ (ROUTE ↔ POINT) → POS
 axm3: Point2InitPos ∈ POINT → POS
 axm4: partition(Pos,{Plus},{Minus})
 axm5: partition(Mode,{Lock},{Unlock})
END
```

B.3.2. Machine

```
MACHINE SR_M2
REFINES SR_M1
SEES SR2_Point
VARIABLES
 ...
 Point2Pos
 Point2Mode
INVARIANTS
 inv1: Point2Pos ∈ POINT → POS
 inv2: Point2Mode ∈ POINT → MODE
 inv3: ∀r,po·po ∈ (Block2Route∼[{r}] ∩ POINT) ∧ r ∈ ran(
     Block2Route) ⇒ (Point2Pos(po) = Route2Point2Pos({r ↦
     po}))
 inv4:∀r,po·po ∈ (Block2Route∼[{r}] ∩ POINT) ∧ r ∈ ran(
     Block2Route) ⇒ (Point2Mode(po) = Lock)
 INITIALISATION:
 THEN
  ...
  act7: Point2Pos := Point2InitPos
```

```
 act8: Point2Mode := POINT × {Unlock}
END

Point_Switch:
ANY r po
WHERE
 grd1: r ∉ Route_Cel
 grd2: r ∉ dom(Route2Path)
 grd3: po ∈ Block2Route∼[{r}] ∩ POINT
 grd4: Point2Pos(po) ≠ Route2Point2Pos({r ↦ po})
 grd5: Point2Mode(po) = Unlock
THEN
 act1: Point2Pos(po) := Route2Point2Pos({r ↦ po})
END

Point_Lock:
ANY r po
WHERE
 grd1: r ∉ Route_Cel
 grd2: r ∉ dom(Route2Path)
 grd3: po ∈ Block2Route∼[{r}] ∩ POINT
 grd4: Point2Pos(po) = Route2Point2Pos({r ↦ po})
 grd5: Point2Mode(po) = Unlock
THEN
 act1: Point2Mode := {po ↦ Lock} ⩤ Point2Mode
END

Point_Unlock_Cancel:
ANY r po
WHERE
 grd1: r ∈ Route_Cel
 grd2: r ∉ Route_Occ
 grd3: po ∈ Block2Route∼[{r}] ∩ POINT
 grd4: Point2Mode(po) = Lock
THEN
 act1: Point2Mode := {po ↦ Unlock}⩤ Point2Mode
END

Point_Unlock_Release:
ANY r cp sp po
WHERE
 grd1: r ∈ Route_Occ
 grd2: po ∈ POINT
 grd3: po ∈ Block2Route∼[{r}]
 grd4: po ∉ Path2Block[{Route2OccPath(r)}]
 grd5: po ∉ Path2Block[{Route2Path(r)}]
THEN
 act1: Point2Mode(po) := Unlock
END

Block_Release:
REFINES Block_Release
 ...
 grd5: b ∈ POINT ⇒ Point2Mode(b) = Unlock
 ...
END

Block_Cancel:
REFINES Block_Cancel
 ...
 grd4: ∀po·po ∈ Block2Route∼[{r}] ∩ POINT ⇒ Point2Mode(
     po) = Unlock
 ...
END

 Route_Reserve:
 ...
 grd6:∀po·(po ∈ (Path2Block[Route2InitPath[{r}]] ∩ POINT))
     ⇒ (Point2Pos(po) = Route2Point2Pos({r ↦ po}))
 grd7:∀po·(po ∈ (Path2Block[Route2InitPath[{r}]] ∩ POINT))
     ⇒ (Point2Mode(po) = Lock)
 ...
END
```

B.4. Third Iteration

B.4.1. Context

```
CONTEXT SR3_Signal
EXTENDS SR2_Point
SETS
 SIGNAL
CONSTANTS
 Red
 Green
AXIOMS
 axm1: partition(SIGNAL,{Red},{Green})
END
```

B.4.2. Machine

```
MACHINE SR_M3
REFINES SR_M2
SEES SR3_Signal
VARIABLES
 ...
 Route2Signal ∈ ROUTE → SIGNAL
 INITIALISATION:
 THEN
  ...
  act9: Route2Signal := ROUTE × {Red}
 END

 Signal_Green_Reserve:
 ANY r
 WHERE
  grd1: r ∈ dom(Route2Path)
  grd2: r ∈ Route_Req
  grd3: r ∉ Route_Cel
  grd4: Route2Signal(r)=Red
 THEN
  act1: Route2Signal := {r ↦ Green} ⩤ Route2Signal
 END

 Signal_Red_Cancel:
 ANY r
 WHERE
  grd1: r ∈ dom(Route2Path)
  grd2: Route2Signal(r) = Green
  grd3: r ∈ Route_Cel
 THEN
  act1: Route2Signal := {r ↦ Red} ⩤ Route2Signal
 END

 Signal_Red_Occupied:
 ANY r
 WHERE
  grd1: r ∈ dom(Route2Path)
  grd2: Route2Signal(r) = Green
  grd3: r ∈ dom(Route2OccPath)
 THEN
  act1: Route2Signal(r) := Red
 END

 Signal_Green_Reserve:
 ANY r
 WHERE
  ...
  grd5: Block2Route∼[{r}] = Path2Block[{Route2InitPath(r)}]
  grd6: r ∉ dom(Route2OccPath)
```

```
50   grd7: ∀po·(po ∈ (Path2Block[Route2InitPath[{r}]] ∩ POINT))
         ⇒ (Point2Pos(po) = Route2Point2Pos({r ↦ po}))
51   grd8: ∀po·(po ∈ (Path2Block[Route2InitPath[{r}]] ∩ POINT))
         ⇒ (Point2Lock(po) = Lock)
52   ...
53 END
54
55 Signal_Red_Cancel:
56 ANY r
57   ...
58   grd4: r ∉ dom(Route2OccPath)
59   grd5: ∀po·(po ∈ (Path2Block[Route2InitPath[{r}]] ∩ POINT))
         ⇒ (Point2Mode(po) = Lock)
60   grd6: ∀po·(po ∈ (Path2Block[Route2InitPath[{r}]] ∩ POINT))
         ⇒ (Point2Pos(po) = Route2Point2Pos({r ↦ po}))
61   grd7: Block2Route∼[{r}] = Path2Block[{Route2InitPath(r)}]
62   ...
63 END
64
65 Signal_Red_Occupied:
66 ANY r
67   ...
68   grd5: Route2OccPath(r) = NullPath
69   ...
70 END
71
72 Point_Unlock_Cancel:
73 REFINES Point_Unlock_Cancel
74   ...
75   grd5: Route2Signal(r) = Red
76   ...
77 END
78
79 Point_Unlock_Release:
80 REFINES Point_Unlock_Release
81   ...
82   grd6: Route2Signal(r) = Red
83   ...
84 END
85
86 Point_Switch:
87 REFINES Point_Unlock_Cancel
88   ...
89   grd7: Route2Signal(r) = Red
90   ...
91 END
92
93 Block_Cancel:
94 REFINES Block_Cancel
95   ...
96   grd5: Route2Signal(r) = Red
97   ...
98 END
99
100 Block_Release:
101 REFINES Block_Release
102   ...
103   grd6: Route2Signal(r) = Red
104   ...
105 END
```